CLINIC MANAGEMENT SYSTEM

A MINI PROJECT REPORT

Submitted by

G K DHAANYA 230701069

T JANANI 230701123

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

THANDALAM

CHENNAI-602105

2024 – 25

BONAFIDE CERTIFICATE

Certified that this project report "**CLINIC MANAGEMENT**" is the bonafide work of **"G K DHAANYA 230701069 , T JANANI 230701123"** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Mrs.Divya.M,	**Mr.Ragu,**
Assistant Professor,	**Assistant Professor,**
Computer Science and Engineering,	**Computer Science and Engineering,**
Rajalakshmi Engineering College,	**Rajalakshmi Engineering College,**
Thandalam,Chennai-602105	**Thandalam,Chennai-602105**

Submitted for the Practical Examination held on_______________________.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT:

In today's fast-paced world, effective management of healthcare services is essential for maintaining high-quality care and improving clinic operations. Small and medium-sized clinics, in particular, often struggle with managing patient records efficiently. Many still rely on manual, paper-based methods or disconnected software, which can lead to mistakes, delays, and inefficiencies. These problems can affect the quality of patient care and the overall productivity of the clinic. To address this, it's crucial to create a Clinic Management System (CMS) that focuses on simplifying patient registration while ensuring data is secure, accurate, and easy to manage.

This project specifically targets the patient registration process, using SQL for the backend database to store and manage patient information, and Java for the frontend interface to make the system user-friendly for clinic staff. The system will allow staff to quickly and securely input patient details such as personal information, medical history, and contact information. By automating the registration process, the system will help reduce errors, save time, and improve the overall patient experience.

Clinics that still use paper-based or outdated software systems often face challenges such as data loss, difficulty accessing patient information, and the potential for human error. This patient registration system is designed to solve these problems by providing a simple, organized, and secure way to register and store patient data. With this system in place, clinics can improve efficiency, reduce administrative burdens, and ensure that patient information is always accurate and accessible when needed.

TABLE OF CONTENTS

Chapter 1 INTRODUCTION

1.1 INTRODUCTION

In the modern healthcare environment, small and medium-sized clinics often face challenges in managing patient records due to outdated or manual systems. This project aims to address these challenges by developing a Clinic Management System (CMS) that focuses on automating and streamlining the patient registration process. The system utilizes SQL for secure and efficient backend data storage, and Java for a user-friendly frontend interface. The primary objective is to simplify patient registration by allowing clinic staff to quickly and accurately input essential patient details, such as personal information, medical history, and contact data. By automating this process, the system reduces the risk of errors, minimizes administrative workload, and improves the overall efficiency of clinic operations. Data security and accuracy are ensured through robust validation features and access controls, while a user-friendly interface ensures quick adoption by clinic staff with minimal training. Additionally, the system is designed to be scalable, allowing for future expansion to include features like appointment scheduling and medical records management. Ultimately, this CMS aims to improve patient care, reduce administrative burdens, and enhance the overall productivity of clinic operations.

1.2 OBJECTIVES

The primary objective of this project is to design and implement a Clinic Management System (CMS) focused on streamlining and automating the patient registration process in a clinic setting. By leveraging SQL for backend data management and Java for the frontend interface, the project aims to achieve the following goals:

-**Simplify Patient Registration**: Create an easy-to-use system that allows clinic staff to efficiently register new patients by entering personal details, medical history, and contact information. The system will automate data entry, reducing the time and effort spent on manual record-keeping.

-**Ensure Data Security and Accuracy**: Implement a secure backend database using SQL to store patient information safely, ensuring that data is accurate, consistent, and accessible only to authorized users. The system will include features like data validation to minimize human error during registration.

-**Improve Clinic Efficiency**: By automating the registration process, the system will eliminate the need for paper-based records, reduce administrative workload, and allow clinic staff to focus more on patient care rather than on paperwork.

-Enhance Data Accessibility: Design a system where patient data can be quickly retrieved, updated, and managed. This will help ensure that patient information is always available and up-to-date, improving overall clinic operations.

-User-Friendly Interface: Develop a simple, intuitive frontend interface using Java that clinic staff can easily navigate. The system will be designed to minimize training time and ensure smooth adoption by clinic personnel.

-Scalability for Future Expansion: Build a flexible and scalable system that can be expanded in the future to include additional features, such as appointment scheduling, medical records management, and billing, if needed.

In essence, the goal of this project is to create a reliable and efficient patient registration system that improves both the accuracy of patient records and the overall productivity of clinic staff.

1.3 MODULES

1 LOGIN

The Clinic Management System (CMS) login page serves as the entry point to the system. Upon accessing the page, users are presented with two login options: one for **Doctors** and one for **Receptionists**. Each user type has specific permissions and functionalities available based on their role within the clinic.

The login interface is simple yet secure, requiring the user to enter their **Username** and **Password** to authenticate their credentials. Once these details are submitted, the system verifies them and grants access to the user's designated dashboard

2 DOCTOR MODULE

Upon logging in, the **Doctor's Page** is tailored to provide the doctor with relevant information about their own professional details and responsibilities. Unlike the receptionist, who has access to patient data, the doctor's dashboard is primarily focused on **viewing and updating their own personal and professional information**, such as their name, role, contact information, and working hours. This is useful for doctors to keep their details accurate and up-to-date within the system.

3 USER CREATION MODULE

The **User Creation Page** is a feature that can only be accessed by the **Receptionist**. This page enables the receptionist to create new user accounts for clinic staff members, including doctors. The user creation process is streamlined, with a series of fields that must be filled in to register the new user within the system.

The receptionist will enter the new user's **Username**, **Password**, **Full Name**, **Role** (e.g., Doctor, Staff), and **Contact Information** (such as phone number). If the new user is a doctor, the receptionist will also specify the specialty, such as General Medicine, Pediatrics, etc. Once the details are entered correctly, the receptionist submits the form, and the new user account is created in the system.

4 PATIENT MODULE

The **Patient Record Page** provides a centralized location for managing patient details, but with different access levels for different users. **Receptionists** have the responsibility of entering and updating patient information, including personal details such as name, phone number etc. Receptionists can also input and maintain accurate contact information for patients, ensuring that the system has up-to-date data. They can search for and access patient records to help schedule appointments and assist with administrative tasks.

Chapter 2 SURVEY OF TECHNOLOGIES

2.1 SOFTWARE DESCRIPTION

The Clinic Management System (CMS) is a secure, user-friendly platform designed to streamline patient registration . Using SQL for backend data storage and Java for the frontend interface, it automates patient record management, improves data accuracy, reduces administrative workload, and enhances overall clinic efficiency and patient care..

2.2 LANGUAGES

2.2.1 SQL

SQL provides a powerful method for organizing, retrieving, and updating large datasets. It enables quick access to patient records, medical histories, and appointments through structured queries. SQL supports operations like filtering, sorting, and joining tables, ensuring efficient data storage and fast data retrieval in real-time.

SQL also maintains data integrity with constraints, preventing errors like duplicates and invalid entries. It supports transactions, ensuring updates are consistent and reliable. This leads to faster operations, reduces delays, and boosts clinic productivity, ultimately enhancing the quality of patient care.

Advantages:

SQL enables efficient management of large datasets, ensuring fast data retrieval, accuracy, and integrity in clinic operations.

2.2.2 JAVA

Java provides a robust and versatile platform for developing the frontend of the Clinic Management System (CMS). It enables the creation of user-friendly, interactive interfaces that make it easy for clinic staff to navigate and manage patient records, appointments, and administrative tasks. Java's object-oriented nature ensures that the system is scalable, allowing for the addition of new features as the clinic's needs grow. Additionally, Java supports secure interactions between the frontend and the SQL backend, ensuring data is accurately and securely transmitted between the user interface and the database. This seamless integration enhances the overall functionality, user experience, and maintainability of the CMS.

Advantages:

Java provides a robust platform for building user-friendly interfaces, ensuring efficient and secure interactions with clinic data while supporting scalability and integration with backend systems.

Chapter 3 REQUIREMENTS AND ANALYSIS

3.1 REQUIREMENT SPECIFICATION

3.1.1 Functional Requirements

User Authentication and Authorization

• User Registration and Login:Allow users to register, create accounts, and log in securely.

• Role-Based Access Control: Assign specific permissions based on user roles (recipient,doctor).

Recipient Registration

The **Recipient Registration** module enables the clinic staff, primarily the receptionist, to create and manage patient profiles. This process involves inputting essential **personal** details such as the patient's full name, date of birth and contact information. This ensures that each patient's record is complete and up-to-date, providing a comprehensive view of the patient's health information for future consultations and treatments.

Receptionists are also able to **update** these profiles as necessary. For instance, if a patient's contact information changes or a new medical condition is diagnosed, the system allows the receptionist to modify the details securely. This functionality ensures that patient information remains accurate and accessible, enhancing the quality of care provided by the clinic. Additionally, the system provides validation checks to reduce the chances of entering incorrect or incomplete data, ensuring that patient records are reliable and consistent.

User Module

The **User Module** is designed to give receptionists the ability to manage the user accounts of clinic staff, such as **admin users**, doctors, and other personnel. Receptionists are responsible for **adding new admins** to the system by creating user profiles that include login credentials, roles, and access permissions.

When a new admin is added, the receptionist can assign specific **roles** (e.g., doctor or receptionist), ensuring each user only has access to the features necessary for their role.

Doctor Module

The **Doctor Module** provides a **comprehensive overview** of the doctor's personal and professional details, allowing doctors to view and update their own information. This includes personal details such as **contact information** and any **specializations** they are associated with, like Pediatrics or General Medicine. The system ensures that this information is readily accessible to the doctor, allowing them to stay organized and up-to-date with their work schedule and responsibilities.

3.1.2 Non-Functional Requirements

Security

- **Data Encryption:** Ensure that sensitive data, including patient information is encrypted while stored in the database to protect it from unauthorized access or breaches.

Performance

- **Scalability:** Design the system to efficiently accommodate increasing numbers of users, patient registrations, and transactions as the clinic grows. The system should be able to handle greater workloads without significant performance degradation.
- **Response Time:** Ensure that the system responds quickly to user actions, such as submitting patient details or querying patient records, with minimal delays to enhance user experience and clinic workflow.

Reliability

- **Availability:** Guarantee high system availability with minimal downtime to ensure that clinic operations are not interrupted and patient care is consistently supported. The system should be robust, with backup mechanisms in place to ensure continuous service.

- **Data Backup:** Implement regular, automated data backups to safeguard against data loss. In case of a system failure, the backup ensures data can be restored quickly and efficiently to minimize disruptions in clinic operations.

Usability

- **User-Friendly Interface:** Provide a simple, intuitive, and easy-to-navigate interface that is tailored for all user roles—receptionists and doctors . The system should require minimal training to use, ensuring seamless adoption by staff members.

Maintainability

- **Modular Design:** Adopt a modular architecture to ensure that the system is easy to maintain, update, and expand. This approach will allow for smooth integration of new features and easier troubleshooting.
- **Comprehensive Documentation:** Provide clear and detailed documentation for both end-users and developers. This will assist users in navigating the system and developers in maintaining and upgrading the platform.

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

Hardware Requirements:

- **Desktop PC or Laptop:** A reliable desktop PC or laptop to run the Clinic Management System (CMS) smoothly, capable of handling everyday clinic operations.
- **Processor:** Intel® Core™ i3 or equivalent, providing sufficient processing power for handling patient data management, appointments, and other tasks without lag.
- **RAM:** 4.00 GB RAM, ensuring the system can handle multiple simultaneous user requests, such as data entry and queries, without performance degradation.
- **System Architecture:** 64-bit operating system with an x64-based processor for optimal performance, supporting the latest technologies and large datasets used by the CMS.
- **Monitor Resolution:** 1024 x 768 monitor resolution, providing a clear, organized interface for clinic staff to interact with the system effectively.
- **Input Devices:** Keyboard and Mouse, enabling easy navigation and user interaction with the system interface.
- **Server:** A server with high processing capacity, sufficient storage, and reliable backup solutions for handling patient records, appointments, and other essential clinic data securely.
- **Reliable Network Infrastructure:** A stable, high-speed network infrastructure to ensure seamless communication between frontend users (doctors,

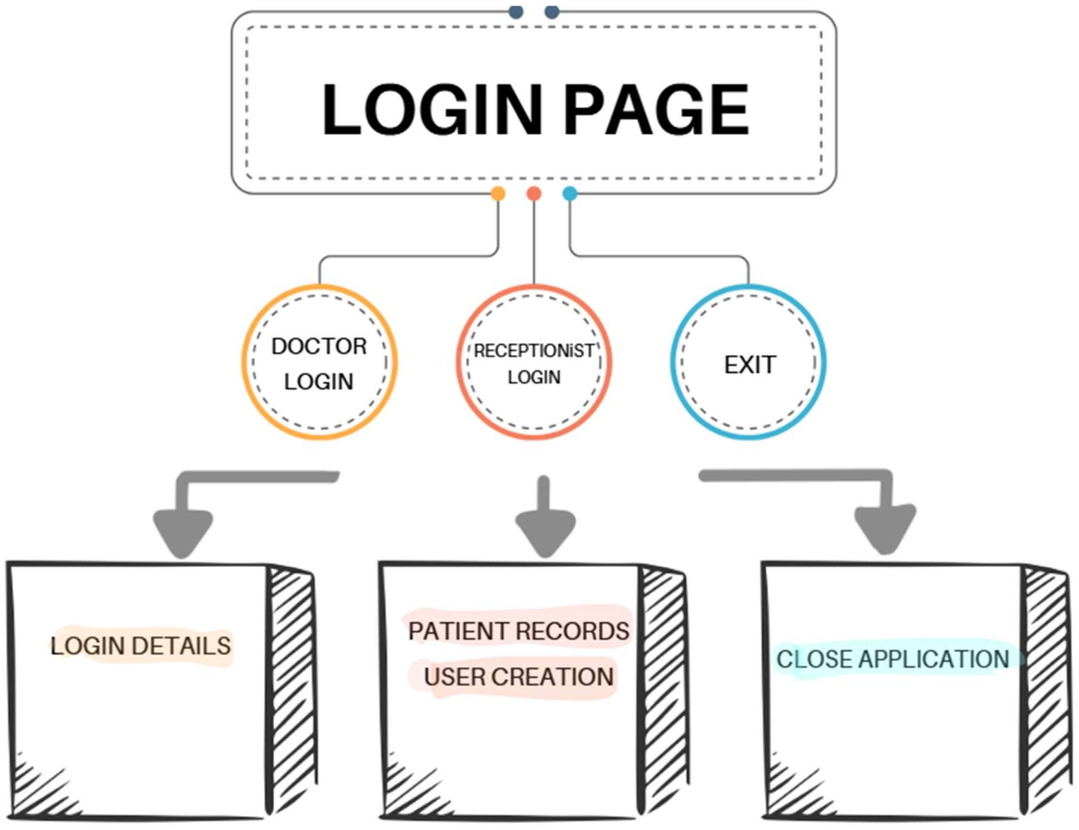
receptionists) and backend systems (server/database).

Software Requirements:

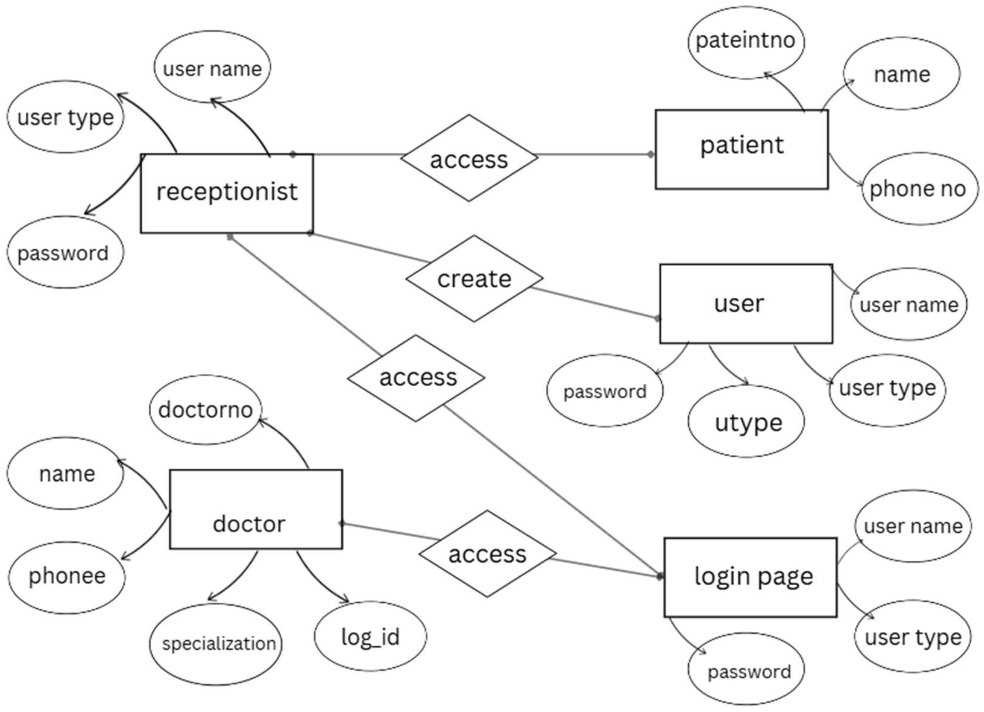
- Operating System: Windows 11
- Code editor : netbeans
- Front End: Java
- Back End: MySQL
- Middleware: XAMPP (Apache, MySQL, PHP

3.3 ARCHITECTURE DIAGRAM

A visual diagram that provides an overall view of the Clinic Management system.



3.4 ER DIAGRAM



3.5 NORMALISATION

DATABASE NAME - jdclinic

Doctor's table

doctorno	name	Phone	specialization	log_id
1	Janani	9048473947	general	IDB654321
2	…..	…..	…..	…..
…..	…..	……	…..	…..

patient's table

patientno	name	phoneno
JD001	harsha	8985739847
JD002	……	……

user's table

name	username	password	utype
Janani	Janani	janani	Doctor
Dhaanya	Dhaanya	dhaanya	Receptionist

CHAPTER4

PROGRAM CODE

1. MAIN PAGE

```
 import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import main.java.User;
    public Main() {
     initComponents();
    }
    int idd;
    String uctype;
    int newid;
    String uname ;
    String usertype;
    public Main(int id,String username,String utype) {
        initComponents();
        this.uname = username;
        jLabel4.setText(uname);
        this.usertype = utype;
        jLabel5.setText(usertype);
        this.newid = id;
        idd= newid;
        uctype = jLabel5.getText();
        if(uctype.equals("Doctor")){
            jButton10.setVisible(false);
            jButton6.setVisible(false);
            }
        else if(uctype.equals("Receptionist")){
            jButton1.setVisible(false);
       }
}
        private void initComponents() {

        jButton8 = new javax.swing.JButton();
        jPanel1 = new javax.swing.JPanel();
        jButton1 = new javax.swing.JButton();
        jButton6 = new javax.swing.JButton();
        jButton9 = new javax.swing.JButton();
        jButton10 = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();
        jPanel2 = new javax.swing.JPanel();
```

```
jLabel2 = new javax.swing.JLabel();

jLabel3 = new javax.swing.JLabel();

jLabel4 = new javax.swing.JLabel();

jLabel5 = new javax.swing.JLabel();


jButton8.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(java.awt.event.ActionEvent evt) {

        jButton8ActionPerformed(evt);

    }

});


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);


jPanel1.setBackground(new java.awt.Color(153, 0, 51));


jButton1.setText("Doctor");

jButton1.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(java.awt.event.ActionEvent evt) {

        jButton1ActionPerformed(evt);

    }

});


jButton6.setText("Create User");

jButton6.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(java.awt.event.ActionEvent evt) {

        jButton6ActionPerformed(evt);

    }

});


jButton9.setText("Exit");

jButton9.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(java.awt.event.ActionEvent evt) {

        jButton9ActionPerformed(evt);

    }

});


jButton10.setText("Patient");

jButton10.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(java.awt.event.ActionEvent evt) {

        jButton10ActionPerformed(evt);

    }

});


javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
```

```
        jPanel1.setLayout(jPanel1Layout);

        jPanel1Layout.setHorizontalGroup(

            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

            .addGroup(jPanel1Layout.createSequentialGroup()

                .addGap(58, 58, 58)

                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)

                    .addComponent(jButton1, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

                    .addComponent(jButton6, javax.swing.GroupLayout.DEFAULT_SIZE, 117, Short.MAX_VALUE)

                    .addComponent(jButton9, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

                    .addComponent(jButton10, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))

                .addContainerGap(56, Short.MAX_VALUE))

        );

        jPanel1Layout.setVerticalGroup(

            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

            .addGroup(jPanel1Layout.createSequentialGroup()

                .addGap(39, 39, 39)

                .addComponent(jButton10, javax.swing.GroupLayout.PREFERRED_SIZE, 35,
javax.swing.GroupLayout.PREFERRED_SIZE)

                .addGap(45, 45, 45)

                .addComponent(jButton6, javax.swing.GroupLayout.PREFERRED_SIZE, 35,
javax.swing.GroupLayout.PREFERRED_SIZE)

                .addGap(45, 45, 45)

                .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 35,
javax.swing.GroupLayout.PREFERRED_SIZE)

                .addGap(45, 45, 45)

                .addComponent(jButton9, javax.swing.GroupLayout.PREFERRED_SIZE, 34,
javax.swing.GroupLayout.PREFERRED_SIZE)

                .addContainerGap(39, Short.MAX_VALUE))

        );


        jLabel1.setBackground(new java.awt.Color(204, 0, 51));

        jLabel1.setFont(new java.awt.Font("Serif", 2, 48)); // NOI18N

        jLabel1.setForeground(new java.awt.Color(204, 0, 0));

        jLabel1.setText("       JD Clinic");


        jPanel2.setBackground(new java.awt.Color(153, 0, 51));


        jLabel2.setBackground(new java.awt.Color(255, 255, 255));

        jLabel2.setFont(new java.awt.Font("Serif", 2, 18)); // NOI18N

        jLabel2.setForeground(new java.awt.Color(255, 255, 255));

        jLabel2.setText(" User Name ");


        jLabel3.setFont(new java.awt.Font("Serif", 2, 18)); // NOI18N

        jLabel3.setForeground(new java.awt.Color(255, 255, 255));

        jLabel3.setText("User Type");
        jLabel4.setFont(new java.awt.Font("Serif", 2, 18)); // NOI18N

        jLabel4.setForeground(new java.awt.Color(255, 255, 255));

        jLabel4.setText("jLabel4");


        jLabel5.setFont(new java.awt.Font("Serif", 2, 18)); // NOI18N

        jLabel5.setForeground(new java.awt.Color(255, 255, 255));

        jLabel5.setText("jLabel5");


        javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);

        jPanel2.setLayout(jPanel2Layout);

        jPanel2Layout.setHorizontalGroup(

            jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

            .addGroup(jPanel2Layout.createSequentialGroup()

                .addGap(40, 40, 40)

                .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

                    .addComponent(jLabel2, javax.swing.GroupLayout.DEFAULT_SIZE, 97, Short.MAX_VALUE)

                    .addComponent(jLabel3, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))

                .addGap(91, 91, 91)

                .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

                    .addComponent(jLabel4)

                    .addComponent(jLabel5))

                .addContainerGap(105, Short.MAX_VALUE))

        );

        jPanel2Layout.setVerticalGroup(

            jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

            .addGroup(jPanel2Layout.createSequentialGroup()

                .addGap(65, 65, 65)

                .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

                    .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 29,
javax.swing.GroupLayout.PREFERRED_SIZE)

                    .addComponent(jLabel4))

                .addGap(28, 28, 28)

                .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

                    .addComponent(jLabel3)

                    .addComponent(jLabel5))

                .addContainerGap(65, Short.MAX_VALUE))

        );


        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

        getContentPane().setLayout(layout);

        layout.setHorizontalGroup(

            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

            .addGroup(layout.createSequentialGroup()
                .addGap(28, 28, 28)

                .addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

                .addGap(18, 18, 18)

                .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

                .addGap(38, 38, 38))

            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()

                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

                .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 339,
javax.swing.GroupLayout.PREFERRED_SIZE)

                .addGap(182, 182, 182))

        );

        layout.setVerticalGroup(

            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

            .addGroup(layout.createSequentialGroup()

                .addContainerGap()

                .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 79,
javax.swing.GroupLayout.PREFERRED_SIZE)

                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

                    .addGroup(layout.createSequentialGroup()

                        .addGap(76, 76, 76)

                        .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))

                    .addGroup(layout.createSequentialGroup()

                        .addGap(18, 18, 18)

                        .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)))

                .addContainerGap(115, Short.MAX_VALUE))

        );


        pack();

        setLocationRelativeTo(null);

    }// </editor-fold>


    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

        // TODO add your handling code here:

        if(uctype.equals("Doctor")){

            String uctype = jLabel5.getText();

            try {

                new Doctor(idd,uctype).setVisible(true);

            } catch (SQLException ex) {

                Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);

            }

        }


    }
    private void jButton8ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void jButton10ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        Patient p = null;
        try {
            p = new Patient();
        } catch (SQLException ex) {
            Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
        }
        p.setVisible(true);
    }

    private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        User u = null;
        try {
            u = new User();
            u.setVisible(true);
        } catch (SQLException ex) {
            Logger.getLogger(Main.class.getName()).log(Level.SEVERE, "error", ex);
        }

    }

    private void jButton9ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
         this.setVisible(false);
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels())
{
```

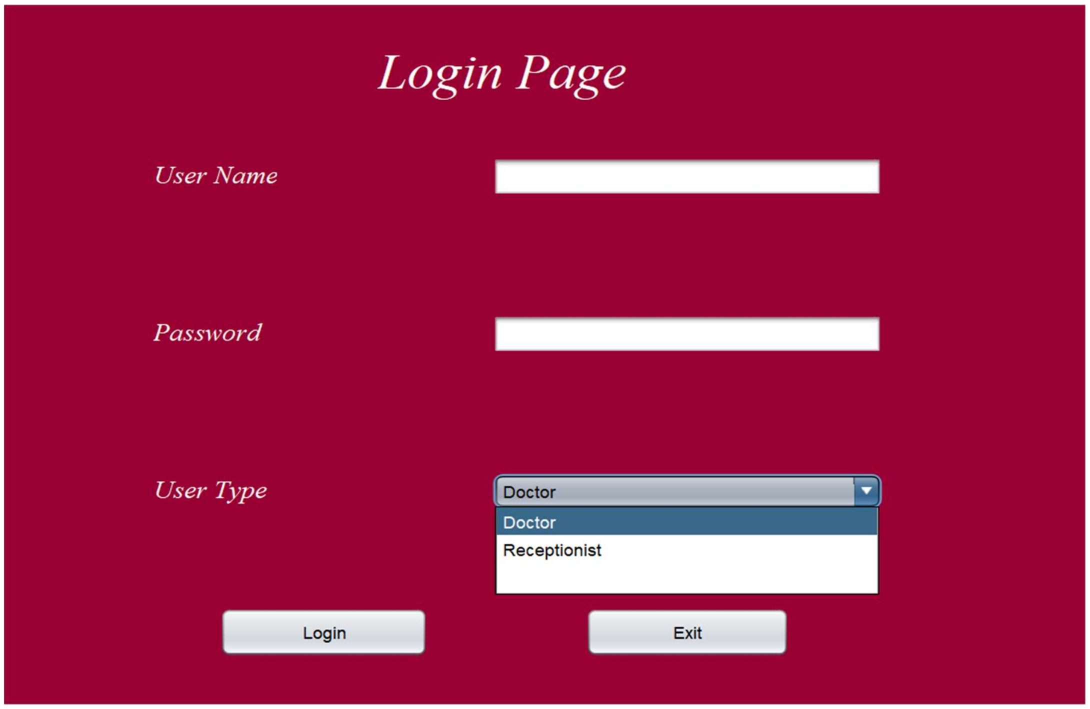
```
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(Main.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(Main.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(Main.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(Main.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Main().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton10;
    private javax.swing.JButton jButton6;
    private javax.swing.JButton jButton8;
    private javax.swing.JButton jButton9;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    // End of variables declaration
}
```

2.LOGIN PAGE

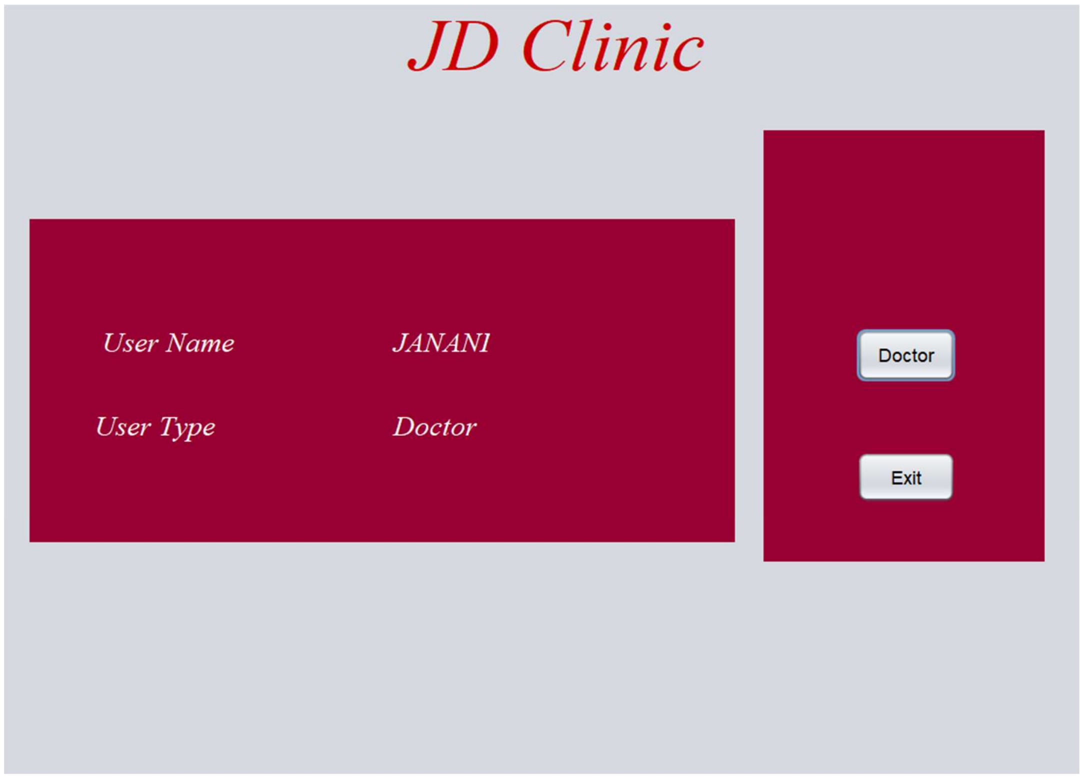
```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import main.java.User;
import java.sql.ResultSet;
import javax.swing.JOptionPane;

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this template
 */

/**
 *
 * @author Janani
 */
public class Login extends javax.swing.JFrame {

    /**
     * Creates new form Login
     */
    public Login() throws SQLException {
        initComponents();
        Connect();
    }
    Connection con;
    PreparedStatement pst;
    ResultSet rs;


    public void Connect() throws java.sql.SQLException{
        try {
            Class.forName("com.mysql.jdbc.Driver");
            con = java.sql.DriverManager.getConnection("jdbc:mysql://localhost/jd clinic","root","");
        } catch (ClassNotFoundException ex) {
            Logger.getLogger(User.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        txtusername = new javax.swing.JTextField();
        txtpassword = new javax.swing.JPasswordField();
        txtutype = new javax.swing.JComboBox<>();
        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();
        jLabel4 = new javax.swing.JLabel();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jPanel1.setBackground(new java.awt.Color(153, 0, 51));
        jPanel1.setBorder(javax.swing.BorderFactory.createCompoundBorder());

        jLabel1.setBackground(new java.awt.Color(255, 255, 255));
        jLabel1.setFont(new java.awt.Font("Serif", 2, 18)); // NOI18N
        jLabel1.setForeground(new java.awt.Color(255, 255, 255));
        jLabel1.setText("User Name");

        jLabel2.setBackground(new java.awt.Color(255, 255, 255));
        jLabel2.setFont(new java.awt.Font("Serif", 2, 18)); // NOI18N
        jLabel2.setForeground(new java.awt.Color(255, 255, 255));
        jLabel2.setText("Password ");

        jLabel3.setBackground(new java.awt.Color(255, 255, 255));
        jLabel3.setFont(new java.awt.Font("Serif", 2, 18)); // NOI18N
        jLabel3.setForeground(new java.awt.Color(255, 255, 255));
        jLabel3.setText("User Type");

        txtusername.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                txtusernameActionPerformed(evt);
            }
        });

        txtutype.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] { "Doctor", "Receptionist", " "
}));

        jButton1.setText("Login");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jButton2.setText("Exit");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        jLabel4.setFont(new java.awt.Font("Serif", 2, 36)); // NOI18N
        jLabel4.setForeground(new java.awt.Color(255, 255, 255));
        jLabel4.setText("Login Page ");

        javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(105, 105, 105)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                            .addComponent(jLabel1)
                            .addComponent(jLabel2, javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(jLabel3, javax.swing.GroupLayout.Alignment.LEADING))
                        .addGap(144, 144, 144)
                        .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                            .addComponent(txtusername)
                            .addComponent(txtpassword)
                            .addComponent(txtutype, 0, 261, Short.MAX_VALUE)))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(149, 149, 149)
                        .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 140,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(105, 105, 105)
                        .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 137,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(255, 255, 255)
                        .addComponent(jLabel4)))
                .addContainerGap(142, Short.MAX_VALUE))
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(27, 27, 27)
                .addComponent(jLabel4)
                .addGap(40, 40, 40)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel1)
                    .addComponent(txtusername, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(86, 86, 86)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(txtpassword, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel2))
                .addGap(86, 86, 86)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel3)
                    .addComponent(txtutype, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 71, Short.MAX_VALUE)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 36,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 36,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(43, 43, 43))
        );

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(21, 21, 21)
                .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addContainerGap())
        );

        pack();
        setLocationRelativeTo(null);
    }// </editor-fold>

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        this.setVisible(false);
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        String username = txtusername.getText();
        String password = txtpassword.getText();
        String utype = txtutype.getSelectedItem().toString();


        try {
            pst = con.prepareStatement("select * from user where username = ? and password = ? and utype =
?");
            pst.setString(1,username);
            pst.setString(2,password);
            pst.setString(3,utype);

            rs = pst.executeQuery();
            if(rs.next()){
                int userid = rs.getInt("id");
                this.dispose();
                this.setVisible(false);
                new Main(userid,username,utype).setVisible(true);
            }
            else{
                JOptionPane.showMessageDialog(this,"UserName or Password do not match ");
                txtusername.setText("");
                txtpassword.setText("");
                txtutype.setSelectedIndex(-1);
                txtusername.requestFocus();
            }








        } catch (SQLException ex) {
            Logger.getLogger(Login.class.getName()).log(Level.SEVERE, null, ex);
        }

    }

    private void txtusernameActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(Login.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(Login.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(Login.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(Login.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    new Login().setVisible(true);
                } catch (SQLException ex) {
                    Logger.getLogger(Login.class.getName()).log(Level.SEVERE, null, ex);
                }
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPasswordField txtpassword;
    private javax.swing.JTextField txtusername;
    private javax.swing.JComboBox<String> txtutype;
    // End of variables declaration
}
```

Chapter 5 RESULTS

LOGIN PAGE



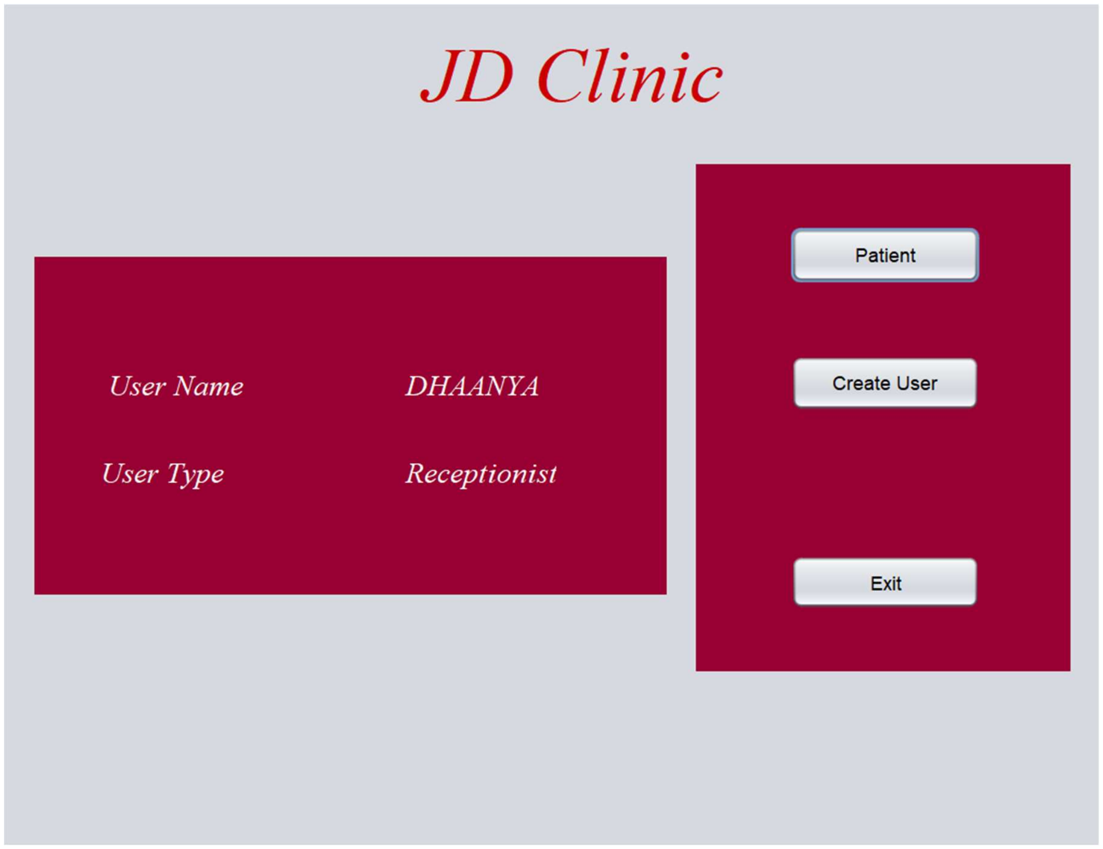
DOCTOR LOGIN



User Name JANANI

User Type Doctor

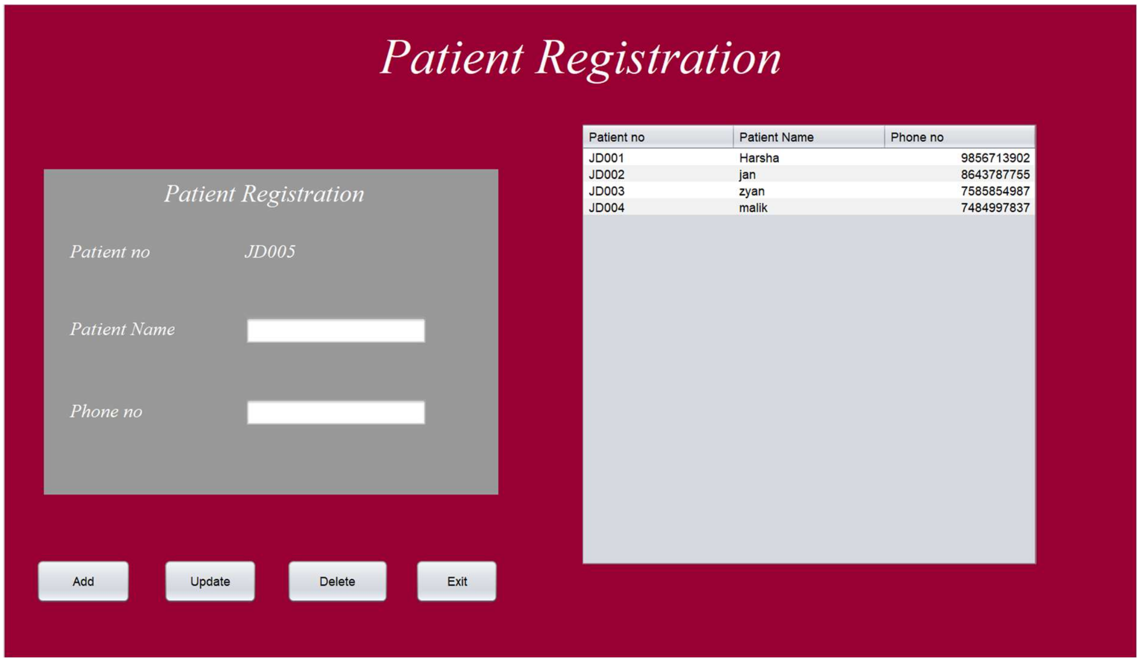
3.RECEPTIONIST LOGIN


JD Clinic

User Name DHAANYA

User Type Receptionist

PATIENT RECORDS


Patient Registration

Patient Registration

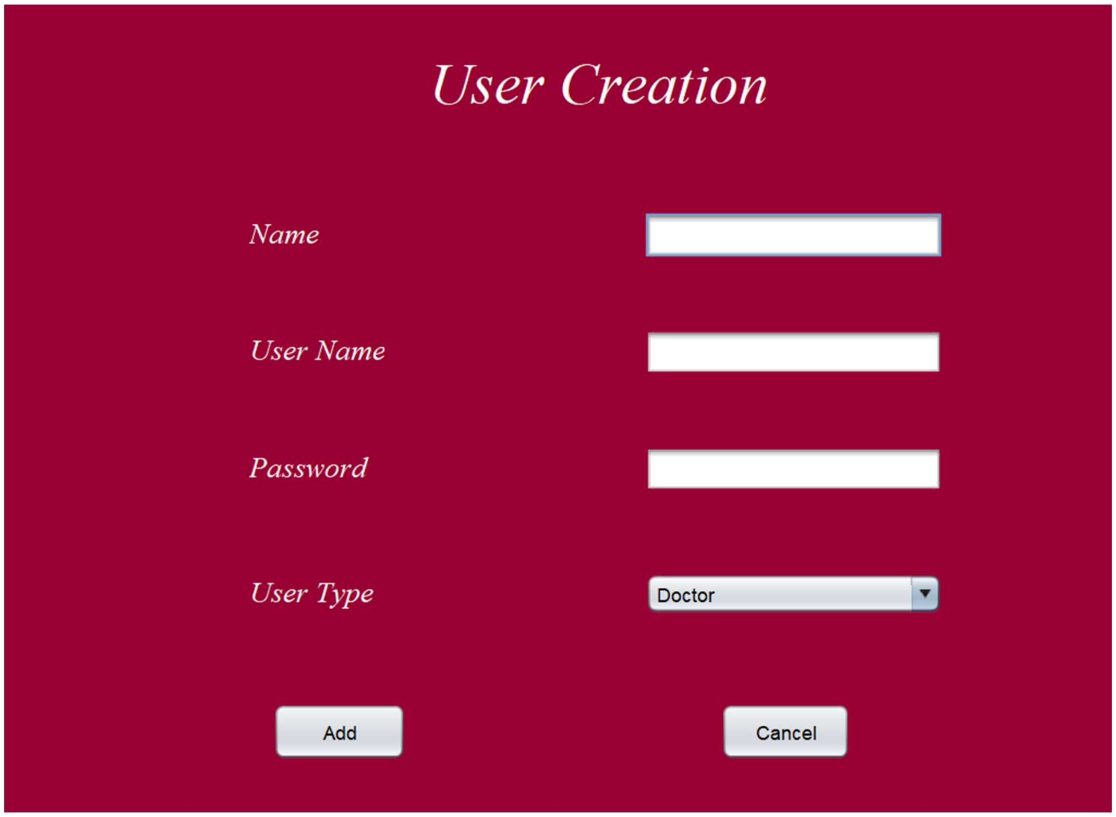
Patient no JD005

Patient Name

Phone no

Patient no	Patient Name	Phone no
JD001	Harsha	9856713902
JD002	jan	8643787755
JD003	zyan	7585854987
JD004	malik	7484997837

USER CREATION



Chapter 6

CONCLUSION

6.1 Conclusion

In conclusion, the Clinic Management System (CMS) developed for this project provides a comprehensive and efficient solution to the challenges faced by small and medium-sized clinics in managing patient records and clinic operations. By automating the patient registration process and streamlining administrative tasks, the system significantly reduces the risk of errors, enhances the accuracy of patient data, and improves overall clinic productivity.

The integration of SQL for backend data management ensures that patient information is stored securely and is easily accessible by authorized personnel, while the Java-based frontend interface provides a user-friendly experience for both receptionists and doctors. With a clear distinction between user roles—such as doctors having access to their own details and medical records, and receptionists managing patient registration and administrative tasks—the system ensures that the appropriate data is available to the right personnel.

Moreover, the system's scalability ensures that it can evolve with the clinic's needs, allowing for the addition of future features such as appointment scheduling, medical records management, and billing. The system’s design prioritizes both **data security** and **ease of use**, making it a reliable tool for improving clinic operations and ensuring better patient care.

Ultimately, this Clinic Management System not only addresses the immediate need for a more efficient, secure, and user-friendly registration process but also lays the foundation for future improvements and expansions within the clinic. With its successful implementation, clinics can expect reduced administrative burdens, more accurate patient records, and an enhanced overall experience for both staff and patients.

Chapter 7

REFERENCES

[1[XAMPP. (n.d.). *XAMPP Documentation*. Retrieved from https://www.apachefriends.org/index.html

[2] MySQL. (n.d.). *MySQL Documentation*. Retrieved from https://dev.mysql.com/doc/

**[3]Java MySQL Connection Tutorial
A step-by-step tutorial on how to connect Java with MySQL using JDBC (Java Database Connectivity).**

[4]Java MySQL Connection Tutorial A step-by-step tutorial on how to connect Java with MySQL using JDBC (Java Database Connectivity).